Web Robot Detection - Preprocessing Web Logfiles for Robot Detection

Christian Bomhardt[1], Wolfgang Gaul[1], and Lars Schmidt-Thieme[2]

[1] Institut für Entscheidungstheorie und Unternehmensforschung
University of Karlsruhe, Germany
[2] Institute for Computer Science
University of Freiburg, Germany

Abstract. Web usage mining has to face the problem that parts of the underlying logfiles are created by robots. While cooperative robots identify themselves and obey to the instructions of server owners not to access parts or all of the pages on the server, malignant robots may camouflage themselves and have to be detected by web robot scanning devices. We describe the methodology of robot detection and show that highly accurate tools can be applied to decide whether session data was generated by a robot or a human user.

1 Introduction

The first web robots appeared in 1993 (The Web Robots Pages): "MOMspider" by Roy T. Fielding (indexing, statistics), "Wanderer" by Matthew Gray (measuring web growth), and "JumpStation" by J. Fletcher (indexing). At these times, most problems of robot deployment appeared in the area of overloaded web servers or waste of bandwidth. Although, in 1993, the web community was "small" compared to nowadays, the increasing use of robots led to the standard for robot exclusion (Koster (1994)). Cooperative robots follow these guidelines and are - in general - easy to detect. Malignant robots ignore these guidelines and may even apply stealth technologies.

Today, one of the most important concerns in web robot detection is unethical content usage (e.g., unauthorized usage of US Government's National Weather Service (NWS) forecast data (Anaconda), extraction of mail addresses for spamming (Ipaopao.com)), and other forms of unexpected usage (bots that sign up email accounts for spamming (Captcha)). Additionally, robot requests decrease web server speed, may distort logfiles (at least 16% of the web traffic originates from robots (Menascé et al. (2000))), and thereby influence serious web mining.

Today's most widely used technologies for robot detection can be divided into four major categories: *Simple methods* (checking the [agent] and [IP address] fields in logfile entries, checking of requests for `robots.txt` (Arlitt et al. (2001)), *traps* (embedding of HTML code that looks like a link, but indeed is invisible for a real user (Mullane (1998))), *web navigation behavior analysis*

(trying to find implicate log characteristics based on the objectives of the different robots (Almeida et al. (2001))) and - as an improvement - *navigational pattern modeling* (defining session attributes and applying data/web mining algorithms that decide in favor of the absence/presence of robot visits based on the calculated attribute values (Tan and Kumar (2000,2001)))). With the simple methods cooperative robots can be detected. The only problem is the actuality of the robot lists as the number of web robots increases and changes of the identification information can occur and have to be updated. This technology is unable to detect malignant robots. Traps can detect malignant robots because trapfile lists can be created with files that would never be requested by human users. If there are such requests they originate from a robot - if not - one cannot be sure whether site visits of robots have occurred. If files from the trapfile list have been requested by an unidentified robot a malignant robot has been found. Web navigation behavior analysis can detect robots - malignant ones as well as new and/or known but modified ones - based on the different ways how human beings and robots access information contained in web sites. A robot detection tool RDT (Bomhardt (2002)) - a specialized web data preprocessing software enabling the researcher to effectively work with and understand large logfile data - one of the main requirements to build accurate prediction models - combines web navigational behavior analysis with navigational pattern modeling.

In the following the robot detection process will be divided into two main phases: *Web Data Preprocessing* with the substeps sessionizing, session labeling, and calculation of session attributes (feature vector) and *Robot Mining* with the substeps robot detection model development and deployment. The robot mining phase is well supported by different software systems like the SAS Enterprise Miner. With the robot detection tool RDT we fill the preprocessing gap and enable researchers to quickly gain accurate input for the robot mining phase. With this support, they can focus on model developing.

2 Web data preprocessing for robot detection

Every webserver can at least write a logfile that lists all HTTP-requests in the order they occur. Each HTTP-request is represented by a single line in the logfile using the combined logfile format (Apache) which most HTTP servers can create. Each logfile entry consists of the following nine fields: [IP address] [name] [login] [date] [request] [status] [size] [referrer] [agent] with [IP address] as client IP address, [name] as name of the user (usually unused), [login] as login-name of the basic HTTP-authentication, [date] as date and time of the request, [request] as HTTP-request containing the request method, the URL of the requested resource (page), and the desired HTTP-protocol, [status] as 3-digit status code returned by the server, [size] as number of bytes actually returned by the server, [referrer] as URL of the referencing page and [agent] as name of the client agent (e.g., "Mozilla/4.75[en](WinNT;U)"). Request,

referrer and agent are provided by the client and are unreliable for analysis. The other fields are generated by the web server and, therefore, trustworthy. The structure of the examined web site is another source of information. It is used for the calculation of advanced session attributes in the later model.

2.1 Sessionizing

The first step - sessionizing - combines single requests (=logfile entries) into user sessions. Berendt et al. (2001) give an overview over the different types of sessionizers and their performance. We use a timeout based heuristics where requests with the same agent and IP address are grouped together as long as the maximum idle time between two requests is smaller than 30 minutes (according to Catledge and Pitkow (1995)). Here, navigation path construction (Gaul and Schmidt-Thieme (2000)) can also be applied. For a common user session, requests can be divided into two groups: main requests as a result of an user action and auxiliary requests automatically issued by browsers to retrieve objects referenced by the main request (images, java applets). The set of requests of one pageview span one main request and its auxiliary requests. We define every requested HTML resource as main request and assign the remaining requests to the main requests corresponding to their referrers. Requests without suitable main request contained in the session are treated as main requests.

2.2 Session labeling

Session labeling describes the operation of assigning a session to a human user or a robot. Robots issued by unregistered users are unable to login into a website as they do not know the necessary username and password. So every session with a login name can be classified as user session. This has to be considered if you run your own administrative robots that use HTTP authentication.

Some files are known to be never requested by real users. These may be some hidden linked files from traps (Mullane (1998)), `robots.txt` requested by robots following the robot guidelines (Koster (1994)) or typical files from worm attacks (e.g., `cmd.exe` for Nimbda (Heng)). All these files are stored in the *trapfile list* and as requests for such files normally originate from a robot, they are used to reliably identify robots.

Cooperative robots that obey to the robot exclusion standard (Koster (1994)) identify themselves with their own agent tag. These tags are contained in the *robot agent list*. The agent field in the request line is sent by the client and malignant robots could use it to camouflage themselves by sending a well known browser agent tag instead of their own. It is therefore impossible to build a "true" user agent list. But it is useful to have a *common agent list* that contains user agents to differentiate between known robot tags, common agent tags or unknown tags.

Some IP addresses are known to be the home of spiders - for example the IP addresses used by the google bots. These IPs can be saved in the *robot IP list* and so requests from those IPs can be identified as robots.

There is a list of known robots available from `http://www.robotstxt.org/wc/robots.html` (The Web Robots Pages). Among other things, it contains the robot name, IP and agent tag. For easier updates, this list is stored separately but used identically to the robot agent and robot IP lists.

You may consider all traffic originating from some special IPs to consist of user sessions (for example, your department's computer room). To achieve this, the tool checks the IP against known IPs from the *user IP list*. We used the session labeling heuristics from figure 1. The algorithm first checks for sessions with given user names followed by the search for requests for files from the trapfile list and the examination of session agent and IP attributes. By default, sessions receive the user label.

```
function LabelSession( Session )
{
if (session contains request with given login name)
   then return user;
if (session contains request for file from trapfile list)
   then return robot;
if (session agent is contained in the robot agent list)
   then return robot;
if (session IP is contained in the robot IP list)
   then return robot;
if (session IP is contained in the user IP list)
   then return user;
return user;
}
```

Fig. 1. Session labeling heuristics

2.3 Calculation of session attributes

By session labeling a large percentage of robot visits can be detected. The next step is the calculation of session attributes (feature vector). Table 1 presents a summary of the attributes calculated. Some attributes (e.g., AVGREQTIME and STDREQDEV need sessions with at least two requests) have individual requirements and may therefore be missing for corresponding sessions. We included the attributes TOTALPAGES, %IMAGE, %HTML, TOTALTIME, AVGTIME, STDEVTIME, %ERROR, GET, POST, HEAD, OTHER, LENGTH, and %NOREFERRER (referrer="-" in Tan and Kumar (2000)) from Tan and Kumar (2000). The AVGREPEATED attribute is a modification of the "repeated" attribute from Tan and Kumar (2000). We left out %BINARY DOC, %BINARY EXEC, %ASCII, %ZIP, %MULTIMEDIA, and %OTHER because the file types of these attributes played a minor role for the examined websites and we think that they are sufficiently considered

within the %HTML and %IMAGE attributes. We excluded the NIGHT attribute because we think that the relevance found in Tan and Kumar (2000) results from well behaving search engine spiders that examine the websites during expected idle times. We do not expect this kind of gentleness from malignant robots and, therefore, left this attribute out. Our experience shows

No.	Name	Description
1	TOTALTIME	session length in seconds
2	LENGTH	number of pageviews
3	TOTALPAGES	total number of requests
4	BYTESSEND	total number of bytes send
5	SITECOVERAGE	$= \frac{min(\text{number of pageviews, number of requested HTML files})}{\text{examined websites total number of HTML pages}}$
6	%HTML	percentage of HTML files requested
7	%IMAGE	percentage of images requested
8	%NOREFERRER	percentage of requests w. o. referrer
9	MAXREPEATED	maximum number of requests per file
10	AVGREPEATED	average number of requests per file
11*	AVGREQTIME	average time between two requests
12*	AVGTIME	average time between two requests within the same pageview
13*	MINAVGVIEWTIME	minimum average time between two requests within the same pageview
14*	MAXAVGVIEWTIME	maximum average time between two requests within the same pageview
15*	STDREQDEV	deviation of the time between two requests
16*	STDEVTIME	average deviation of the time between two requests within the same pageview
17*	MINSTDEV	minimum deviation of the time between two requests within the same pageview
18*	MAXSTDEV	maximum deviation of the time between two requests within the same pageview
19	%STATUS200	percentage of requests with status code 200 (OK)
20	%STATUS2XX	percentage of requests with status code 2XX (without 200)
21	%STATUS301	percentage of requests with status code 301 (moved permanently)
22	%STATUS302	percentage of requests with status code 3o2 (moved temporarily)
23	%STATUS304	percentage of requests with status code 3o4 (not modified)
24	%ERROR	percentage of requests with all other status codes (server- or client- error)
25*	%AVGHTMLLINKCOV	average percentage of visited HTML links per requested HTML page
26*	%MINHTMLLINKCOV	minimum percentage of visited HTML links per requested HTML page
27*	%MAXHTMLLINKCOV	maximum percentage of visited HTML links per requested HTML page
28*	%AVGDIVLINKCOV	average percentage of visited non-HTML links per requested HTML page
29*	%MINDIVLINKCOV	minimum percentage of visited non-HTML links per requested HTML page
30*	%MAXDIVLINKCOV	maximum percentage of visited non-HTML links per requested HTML page
31	GET	percentage of requests made with the GET method
32	HEAD	percentage of requests made with the HEAD method
33	POST	percentage of requests made with the POST method
34	OTHER	percentage of requests made with other methods

(*=cannot be calculated for every session)

Table 1. Session Attributes. Gray columns correspond to attributes not found in Tan and Kumar (2000,2001).

that modern spiders no longer seem to apply breadth-first search but instead use multiple non-contiguous spider sessions originating from different IPs (cp. the google bots).

The WIDTH and DEPTH parameters were replaced by a set of similar link coverage (LINKCOV) attributes (attributes no. 25-30). The consideration of link coverage parameters calculated for two pages can be difficult if one page contains two linked pages and the other 200. Therefore we added a minimum constraint for the consideration of pages within the calculation of these attributes. If a page contains less links than the minimum constraint, this page is sorted out and not considered for the calculation of link coverage parameters. In doing so, the parameters were calculable for most pages without a too strong influence on important link coverage attributes (especially the %MINDIVLINKCOV and %MINHTMLLINKCOV parameters).

3 Improvement of data quality by RDT

Improvement of data quality is used as generic term for activities as data cleansing, (re)structuring, and calculating derived information. The mass of web data forces the usage of mining tools for such tasks. Our web data preprocessing software RDT aims at supporting the improvement of data quality in connection with web robot detection. During our preprocessing work, we identified the following problems for which we were able to suggest improvements. All of them are addressed by our robot detection tool RDT.

Improvement of acquisition of derived information: The session labeling heuristics relies on the trapfile, robot agent, robot IP, and user IP lists. These lists usually contain many similar entries (`trap1.html` and `trap2.html`, `Java1.1` and `Java1.1.8`, `crawler10.googlebot.com` and `crawler11.googlebot.com`, `172.22.82.151` and `172.22.82.152`). Administration can be highly improved by introducing regular expressions instead of full qualified expressions. Unknown user agents or clumsy selected regular expressions could lead to misclassifications caused by the robot agent or common agent list. This problem has been addressed by a robot detection tool function that alphabetically sorts all agents found in a logfile in one of three lists: unknown agents, robot agents, and common agents. This enables the researcher to quickly overlook the background knowledge classification quality in the area of the session agent field analysis. The robot detection tool RDT helps the user to modify the robot and common agent lists while viewing the classification result. Changes are immediately incorporated in the classification. This user-friendly approach enables the researcher to inspect the analysis results of several thousand different user agents. For convenience, the list of webrobots can be downloaded from the web and imported into the robot detection tool.

Improvement of knowledge about site structure information: Some attributes in the feature vector (e.g. %AVGHTMLLINKCOV, %AVGDIVLINKCOV) rely on information about the site structure. Thus, a site spider was devel-

oped as part of our robot detection tool RDT which collects information for the calculation of the set of link coverage parameters.
Improvement of data understanding: Data understanding is important when the information basis is huge and difficult to survey as for web mining. Logfiles are hard to read for humans as they simply log requests successively. It is difficult to see which requests belong to a specific session or pageview. For inspections of the navigation, the set of links contained in one page should be handy available. The robot detection tool RDT incorporates requirements of this kind by offering a site structure explorer that shows all links contained in the underlying web site and calculates site structure statistics (minimum, maximum, average and standard deviation of number of HTML links contained in each page and corresponding parameters for non-HTML links). Requests belonging to the same session are written successively in the output log file. Optionally, sessions are separated by empty lines in the output log. The tool has an interactive mode where it displays every session found together with the contained requests, grouped into pageviews, and the calculated session attributes as session detail view. The tool also offers the possibility to access the lists holding background knowledge for the session labeling heuristics. Another option is that unknown agents may belong to some new kind of robot or browser and that it is worth taking a closer look at such a session. The robot detection tool RDT supports this by the option to display the session detail view for sessions with unknown agents. It is also obviously useful to show session details for those sessions with contradicting heuristics and prediction model results. The robot detection tool RDT enables this by providing a plug-in interface for prediction models. One can easily integrate a generated C score code from data mining tools like the SAS Enterprise Miner into the tool. By using this plug-in interface together with handcrafted decision rules, the tool can be used as flexible preprocessing support for other web mining applications like "filter all sessions with at least 4 pageviews". A session of a client using a non-robot agent but requesting a file from the trapfile list will be detected and presented to the researcher.

4 Robot mining

Prediction models using logistic regression, neural networks, and decision trees were applied for robot mining. Sessions containing only a single request show many missing or constant values in their feature vector (TOTALPAGES, LENGTH, SITECOVERAGE, MAXREPEATED, AVGREPEATED, AVGREQTIME, AVGTIME, MINAVGTIME, MAXAVGTIME, STDREQDEV, STDEVTIME, MINSTDEV, MAXSTDEV), as several attributes require at least two requests for calculation. Therefore, we worked with three different datasets: *all sessions*, *single-request-sessions*, and *2-or-more-requests-sessions*. We did not - like Tan and Kumar (2000) - generate a model for every number of pageviews as we worried about overfitting and too small

datasets. The models built from all sessions were used as baseline for the evaluation of models combining single-request- and 2-or-more-requests-situations.

The models were evaluated using standard metrics as misclassification rate, recall and precision. Let S be the set of all sessions, $c : S \rightarrow \{0,1\}$ a map assigning the true class label to each session (1 = robot, 0 = user) and $\hat{c} : S \rightarrow \{0,1\}$ a map assigning the predicted class label to each session (for a given prediction model). Then *misclassification rate* is defined as

$$\text{mis} := \frac{|\{s \in S \,|\, c(s) \neq \hat{c}(s)\}|}{|S|},$$

recall as

$$\text{rec} := \frac{|\{s \in S \,|\, c(s) = \hat{c}(s) = 1\}|}{|\{s \in S \,|\, c(s) = 1\}|},$$

and *precision* as

$$\text{prec} := \frac{|\{s \in S \,|\, c(s) = \hat{c}(s) = 1\}|}{|\{s \in S \,|\, \hat{c}(s) = 1\}|}.$$

For different datasets the misclassification rate can reasonably be compared only, if one takes into account its data set specific baseline value. The baseline value is the value that can be achieved by trivial models that always predict the label of the larger class. Let $m \in \{0,1\}$ be the majority class label, usually the user class label. Then

$$\text{mis}^{\text{base}} := \frac{|\{s \in S \,|\, c(s) = 1 - m\}|}{|S|}.$$

5 Empirical results

For the evaluation of the robot detection tool RDT, we examined a logfile from an educational website (EDU) and another one from a medium sized online shop (SHOP). The EDU logfile had 790142 requests. 2905 different agents could be recognized. 58009 sessions where constructed from the raw data with 28534 identified robot sessions. The single-request-sessions dataset had a volume of 25573 sessions with 26,23% user sessions (which can be a hint that robots use "short" visits and return after session time-out has occurred). The 2-or-more-requests-sessions dataset had a volume of 32436 sessions with 70,19% user sessions. The SHOP logfile contained 1150827 requests from which 52295 sessions could be built. 14068 of these sessions were robot sessions. Correlation analysis showed that our feature vector contained good prediction variables for both logfiles (cp. figure 2). Some attributes were strongly correlated with session labeling for both datasets (%HTML, %IMAGE, %NOREFERRER, AVGREQTIME, %AVGHTMLLINKCOV, %MINHTMLLINKCOV, %MAXHTMLLINKCOV, %AVGDIVLINKCOV, %MINDIVLINKCOV, %MAXDIVLINKCOV) while others (STDREQDEV, %STA-

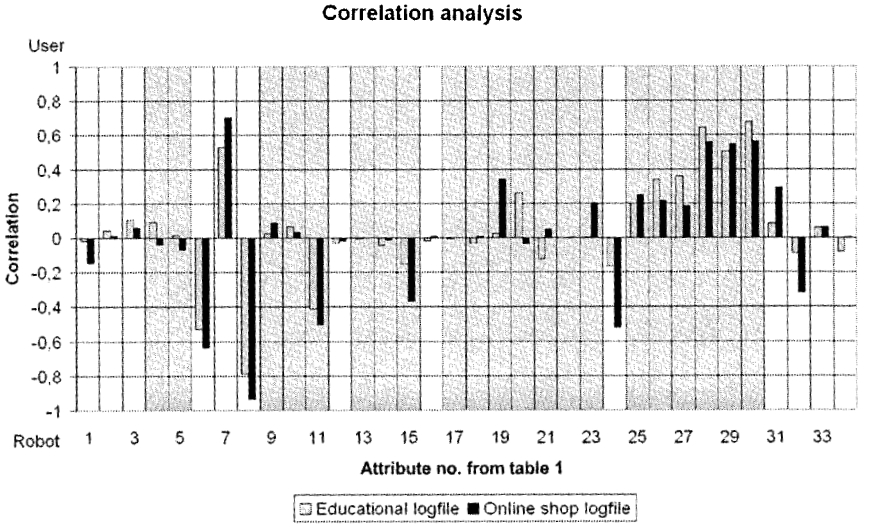

Fig. 2. Correlation analysis. Gray columns correspond to attributes not found in Tan and Kumar (2000,2001).

TUS200, %STATUS2XX, %STATUS304, %ERROR, HEAD) are only essential to one of the two logfiles. For the EDU logfile, the attributes %MAXDIVLINKCOV, %AVGDIVLINKCOV, %MINDIVLINKCOV, %MAXHTMLLINKCOV, %AVGHTMLLINKCOV, %STATUS2XX, and %MINHTMLLINKCOV are very important. Especially the correlations of %MAXDIVLINKCOV with session labeling (stronger than %IMAGE (one of the best attributes from Tan and Kumar (2001))) show that this attribute has to be taken into consideration because typical user sessions should have a high average value for %MAXDIVLINKCOV (as an result of hitting a web page containing only non-HTML links to automatically loaded images). On the other hand, %IMAGE only reaches 100 for user sessions if they download multimedia files linked by external sites without hitting a HTML site. This situation is rare in contrast to robot sessions solely requesting multimedia files because most search indexes have specialized spiders for multimedia content that only examine multimedia files and not download any HTML file (e.g., "FAST-WebCrawler/2.2.10 (Multimedia Search)"). For the SHOP logfile, %IMAGE and %HTML are the strongest attributes followed by the set of link coverage attributes. The correlation of %ERROR, %STATUS200 (OK), and %STATUS304 (not modified) with session labeling shows that user sessions contain primarily requests with the status code "OK" or "not modified".

Very precise models can easily be build using the network part of the IP address or the %NOREFERRER attribute, as most robot traffic originates

from "friendly" search index robots. They do not use any stealth technologies and therefore issue an empty referrer field and periodically respider the site from the same IP address range. Malignant robots are very unlikely to use the same IP more than once and easily issue fake referrers. This is why we ignored corresponding attributes for model building.

For logistic regression and neural networks algorithms missing values where replaced by the mean value of the corresponding attribute. Decision tree algorithms can handle missing values intrinsically, so no imputation was performed. Missing values within the attributes AVGTIME, MINAVGTIME, MAXAVGTIME, STDEVTIME, MINSTDEV, and MAXSTDEV could appear as a recoding of the %NOREFERRER attribute as those attributes are among other things not calculable due to missing referrers. We examined this suspicion. Including the %NOREFERRER field improved the misclassification rates by 5% to 8%. On the other hand excluding the potentially recoded attributes resulted in about 0.2% increased misclassification rates showing that the suspicion was not justified.

The datasets where split into 40% training data, 20% validation data and 40% test data for model building.

For the EDU logfile, table 2 shows the baseline misclassification rate together with the mis, rec, and prec metrics calculated for the test datasets of all sessions, the single-request- and the 2-or-more-requests-sessions. For a combination of the best model for the single-request-sessions dataset and the 2-or-more-requests-sessions dataset, we calculated a mis value of 0.0853%. During our research, we identified several typical kinds of robot and user sessions for this website (e.g. survey attendees, single pageview visits originating from search indexes, robot sessions with an extremely low or high number of requested pages). This is a strong hint that navigational patterns do exist and that they can be used as an indicator for distinguishing between user and robot sessions.

Table 3 shows the results for the SHOP logfile. For a combined model, we calculated a mis value of 0.0654%. Again, we identified different typical kinds of user and robot sessions for this website. Additionally, we checked a dataset consisting solely of sessions with at least 3 pageviews. The generated prediction model, a neural network, achieved a mis value of 0.0157%, a rec value of 0.947 and a prec value of 0.954. For 3 or more pageviews, Tan and Kumar (2001) achieved (on a different dataset) precision above 82% and 95% recall, respectively.

6 Conclusions and outlook

The developed robot detection tool RDT enormously speeds up the preprocessing step within the overall web mining task. It's features enable researchers to efficiently produce high quality input data for the robot mining algorithms. The selected feature vector together with the low noise input

No.	Dataset	model	mis^{base}	mis	rec	prec
1	all sessions	logistic regression.	0.4919	0.1196	0.889	0.871
2	all sessions	neural network	0.4919	0.089	0.924	0.897
3	all sessions	decision tree	0.4919	0.0871	0.938	0.891
4	2-or-more-requests-sessions	logistic regression	0.2981	0.0527	0.920	0.908
5	2-or-more-requests-sessions	neural network	0.2981	0.0472	0.927	0.919
6	2-or-more-requests-sessions	decision tree	0.2981	0.0486	0.916	0.924
7	single-request-sessions	logistic regression	0.2623	0.1636	0.940	0.854
8	single-request-sessions	neural network	0.2623	0.1419	0.931	0.883
9	single-request-sessions	decision tree	0.2623	0.1338	0.962	0.871
10	all sessions	combination of 5 & 9	0.4919	0.0853	0.942	0.898

Table 2. Educational logfile: prediction quality

No.	Dataset	model	mis^{base}	mis	rec	prec
1	all sessions	neural network	0.2690	0.0721	0.850	0.880
2	all sessions	decision tree	0.2690	0.0703	0.832	0.901
3	2-or-more-requests-sessions	neural network	0.2013	0.0242	0.948	0.933
4	2-or-more-requests-sessions	decision tree	0.2013	0.0259	0.950	0.924
5	single-request-sessions	neural network	0.4477	0.1945	0.681	0.850
6	single-request-sessions	decision tree	0.4477	0.1743	0.779	0.818
7	all sessions	combination of 3 & 6	0.2690	0.0654	0.902	0.901
8	3 or more pageviews	neural network	0.1592	0.0157	0.947	0.954

Table 3. Online shop logfile: prediction quality

data lead to highly accurate prediction models. As expected, the generated models depend on the examined web site and confirm our decision to support robot mining by web data preprocessing devices as best fitting models have to be generated for every web site.

A methodological shortcoming of all approaches to robot mining so far is the usage of sessions as underlying object structure: first, sessions are built, usually by making use of behavioral parameters as an empirical session timeout, then, in a second step, prediction models for sessions are constructed. As robot sessions may differ considerably in parameters used for session building, e.g., the session timeout, a two stage approach could further improve prediction quality as well as conceptual understanding of the data: at the first stage, a model for predicting crucial session building parameters (as timeout) is learned, then, at the second stage, sessions are built using the dynamically predicted session parameters from the first stage and analyzed by a prediction model for sessions as before. We will address this issue in a forthcoming paper.

A second open problem that robot mining and web usage mining have in common is the adequate handling of dynamically created pages.

References

ALMEIDA, V., RIEDI, R., MENASCÉ, D., MEIRA, W., RIBEIRO, F., and FONSECA, R. (2001): Characterizing and modeling robot workload on e-business sites. *Proc. 2001 ACM Sigmetrics Conference.* `http://www-ece.rice.edu/riedi/Publ/RoboSimg01.ps.gz`.

ANACONDA partners llc: Anaconda! foundation weather. `http://anaconda.net/ap_wxdemo.shtml`.

APACHE http server documentation project: Apache http server log files combined log format. `http://httpd.apache.org/docs/logs.html\#combined`.

ARLITT, M., KRISHNAMURTHY, D., and ROLIA, J. (2001): Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology.* `http://www.hpl.hp.com/techreports/2001/HPL-2001-110R1.pdf`.

BERENDT, B., MOBASHER, B., SPILIOPOULOU, M., and WILTSHIRE, J. (2001): Measuring the accuracy of sessionizers for web usage analysis. *Proceedings of the Web Mining Workshop at the First SIAM International Conference on Data Mining, Chicago.*

BOMHARDT, C.(2002): The robot detection tool. `http://www.bomhardt.de/bomhardt/rdt/produkt.html`.

CAPTCHA project: Telling humans and computers apart. `http://www.captcha.net/`.

CATLEDGE, L. and PITKOW, J. (1995): *Characterizing browsing strategies in the World-Wide Web. Computer Networks and ISDN Systems.*

GAUL, W. and SCHMIDT-THIEME, L. (2000): Frequent generalized subsequences - a problem from webmining. In: Gaul, W., Opitz, O., Schader, M. (eds.): *Data Analysis, Scientific Modelling and Practical Application*, Springer, Heidelberg, pp. 429-445.

HENG, C.: Defending your web site / server from the nimbda worm / virus. `http://www.thesitewizard.com/news/nimbdaworm.shtml`.

IPAOPAO.COM software Inc.: Fast email spider for web. `http://software.ipaopao.com/fesweb/`.

KOSTER, M. (1994): A standard for robot exclusion. `http://www.robotstxt.org/wc/norobots-rfc.html`.

MENASCÉ, D., ALMEIDA, V., RIEDI, R., RIBEIRO, F., FONSECA, R., and MEIRA, W. (2000): In search of invariants for e-business workloads. *Proceedings of ACM Conference on Electronic Commerce, Minneapolis, MN.* `http://www-ece.rice.edu/riedi/Publ/ec00.ps.gz`.

MULLANE, G. (1998): Spambot beware detection. `http://www.turnstep.com/Spambot/detection.html`.

TAN, P.-N. and KUMAR, V. (2000): Modeling of web robot navigational patterns. *Proc. ACM WebKDD Workshop.*

TAN, P.-N. and KUMAR, V. (2001): Discovery of web robot sessions based on their navigational patterns. `http://citeseer.nj.nec.com/443855.html`.

THE WEB ROBOTS PAGES. `http://www.robotstxt.org/wc/robots.html`.